IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

GENERIC SECURITY CLAIM PROCESSING MODEL

INVENTORS:

GIOVANNI M. DELLA-LIBERA
VIJAY K. GAJJALA
TOMASZ JANCZUK
JOHN R. LAMBERT
ELLIOT WAINGOLD

ATTORNEY'S DOCKET NO. MS1-1858US

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to U.S. Patent Application (Attorney Docket No. MS1-1857US), filed February 16, 2004, entitled "Security Scopes and Profiles".

FIELD

[0002] Various embodiments described below relate generally to security mechanisms for computing environments, more particularly but not exclusively to, authentication and authorization of messages.

BACKGROUND

[0003] Many message-based computing systems include security scheme for messages sent from one process (e.g. a piece of running software) to another process. Typically, such security schemes include an authentication mechanism in which the sender's "identity" is verified (e.g., checking a username/password) and an authorization mechanism in which the "actions" (e.g., accessing a resource) the sender is authorized to perform are determined. An access control mechanism can then be used to determine whether the message can then be allowed to proceed to the target process.

[0004] However, conventional message-based systems typically support only one security mechanism with a single level of security. Consequently, when a message is sent in a path having multiple security schemes (e.g., when the path has one or more intermediaries that use a different security scheme from that of the original sender), the security process can become complex (e.g., requiring the original sender to know the security scheme of each intermediary so that the message will meet the security requirements).

SUMMARY

[0005] In accordance with aspects of the various described embodiments, a system for processing multiple types of security schemes is provided. The system includes a server having a claims engine that extracts claim(s) from security token(s) and maps extracted claims to other claims. The term "claim" as used in this context is a statement about a token's subject. The claims engine can be selectively configured to extract claim(s) from one or more different types of security tokens corresponding to the multiple security schemes. The extracted claim(s) can then be selectively mapped to other claims. The security decision can then be based on the extracted and/or derived claim(s) rather than tokens. This aspect allows a system to support multiple security schemes and simplifies the security process, thereby providing a generic solution.

[0006] In another aspect, each claim is associated with a resource, and the security decision is to allow/deny access to the resource. The claims engine can be configured to determine the resource(s) being accessed by extracting or obtaining resource identifiers from a message at run-time (e.g., a property of the runtime environment), or by examining the static configuration of the service (e.g., a default resource). This aspect increases the flexibility/extensibility of the system.

[0007] In yet another aspect, the claims engine can send a return message to a sender of the message, with the return message including derived claims and/or information related to the location or accessing of derived claims. The sender can then send the derived claims or information pertaining to the derived claims in a subsequent message. This aspect eliminates the need for claim mapping on those subsequent messages, thereby reducing the amount of processing performed by the claims engine.

[0008] In still another aspect, the claims engine can be configured to selectively reject a claim (e.g., the server does not trust certificates from a particular certificate authority). This aspect increases the flexibility of the system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Non-limiting and non-exhaustive embodiments are described with reference to the following figures, wherein like reference numerals refer to like parts throughout the various views unless otherwise specified.

[0010] FIG. 1 is a block diagram illustrating a system using a generic security claim processing model according to one embodiment.

[0011] FIG. 2 is a flow diagram illustrating operational flow in the system of FIG. 1 in processing security claims, according to one embodiment.

[0012] FIG. 3 is a block diagram illustrating exemplary modules of a claims engine, according to one embodiment.

[0013] FIG. 4 is a flow diagram illustrating data flow in the claims engine of FIG. 3, according to one embodiment.

[0014] FIG. 5 is a block diagram illustrating an example computing environment suitable for practicing the above embodiments.

DETAILED DESCRIPTION

[0015] FIG. 1 is a block diagram illustrating a message-based system 100 that uses a generic security claim processing model for security processing of messages, according to one embodiment. In this example, system 100 includes a client 102 and a server 104. Client 102 and server 104, for example, can be different processes executing on a single computing platform, different units of code within the same process, or

different nodes of a network, Client 102 includes an application 106 and a security module 108. Server 104 includes an application 112 and a security module 114. In accordance with this embodiment, security module 114 includes a claims engine 116 that implements the generic security claim processing model. The operation of claims engine 116 is described below in conjunction with FIG. 2.

[0016] FIG. 2 illustrates operational flow in claims engine 116 of system 100 (FIG. 1), according to one embodiment. Referring to FIGS. 1 and 2, system 100 processes a message as follows. Application 106 of client 102 generates a message to be sent to server 104. Security module 108 can then selectively perform security operations (e.g. encryption) of all or part of the message and can then add signed or unsigned security token(s) (e.g., username/password, X.509 certificates, Kerberos tickets, etc.) to the message. For example, the token can be a Kerberos ticket conforming to Kerberos release krb5-1.3.1, October 24, 2003 (also referred to herein as the Kerberos Standard), and/or a X.509 certificate conforming with Internet X.509 Public Key Infrastructure Certificate Management Protocols, April 15, 2003 (also referred to herein as the X.509 Standard). In some embodiments, the token(s) are sent "out-of-band" from that of the message. The token(s) typically include one or more claims, but may have no claims in some scenarios. The claim(s) that result from the mapping process are typically associated with a resource referred to within the message. For example, the resource may be identified by content in the message, determined at run-time, or a default resource that can be accessed by claims engine 116. The message and token(s) are then sent to server 104 (e.g., over a network). In some embodiments, the message and token(s) are sent using an eXtensible Markup Language (XML)-based protocol such as Simple Object

Access Protocol (SOAP) version 1.2, W3C Recommendation 24 June 2003. In other embodiments, any suitable messaging or communication protocol can be used.

[0017] In a block 202, claim(s) are extracted from the token(s). In this embodiment, claims engine 116 of security module 114 extracts the claim(s). As used herein, a claim is a statement about the subject of the security token asserted by either the subject itself or by another party about the subject and, as previously described, associated with a resource. There may be more than one claim associated with a resource. In one embodiment, a claim can be in the form of a "type" and a "value". For example, a claim extracted from a token can have a type of "X.509 Subject" and a value of "CN=John Doe, E=JohnDoe@xyz.com". Claims can also be used to assert a role assumed by or assigned to the subject of the token (e.g., the type can be "role" and the value can be "customer"). Claims can also be used to provide information about cryptographic keys owned by the subject, which also may have been used to encrypt or sign information contained in the message.

[0018] In a block 204, the extracted claim(s) are mapped to other claim(s). In this embodiment, claims engine 116 maps the extracted claim(s) into other claim(s), also referred to herein as "derived" claim(s). In some scenarios, claims engine 116 may not have mappings from the extracted claim(s) to other claim(s). Further, in some embodiments, the mapping definitions may include revocation of preselected claim(s). In one embodiment, these mappings are defined during configuration of server 104. For example, mapping(s) can be defined in a configuration file, in code, in scope and/or profiles (e.g., as described in aforementioned U.S. Patent Application [Attorney Docket MS1-1857US] entitled "Security Scopes and Profiles"). Block 204 may be performed

multiple times to ensure that all of the possible valid claim mapping(s) are found. For example, block 204 can be repeated until the derived claim(s) remain unchanged.

[0019] In a block 206, the extracted and derived claim(s) resulting from block 204 are collected so that they can be used in further processing. In this embodiment, these claim(s) are stored in-memory, but in other embodiments the resulting claim(s) can be stored in a file, database, etc.

[0020] In a block 208, the claim collection is used to allow/deny access to the resource(s) referred to in the message. In this embodiment, claims engine 116 tests the claim(s) to determine whether to allow/deny access, unlike conventional systems in which the access control decision is based on the token or other information specific to the message sender. This scheme allows for multiple security mechanisms to be used in the message path that has intermediaries and does not depend on the messaging system.

[0021] Although the above operational flow is described sequentially, in other embodiments, the operations described in the blocks may be performed in different orders, multiple times, and/or in parallel.

EXAMPLE SCENARIO

[0022] For example, employee John Doe of XYZ Company wishes to purchase an item that is available from an "on-line" retailer for work-related purposes. XYZ Company in this example has created an entity (i.e., an intermediary) through which employees authorized to make "Internet purchases" may request items via an intranet and the company will then order and pay for the item over the Internet. Employee John Doe can send the request (which includes the Universal Product Code or UPC of the desired item) via client 102 to the intermediary (i.e., server 104 in this example) along with, for

example, a Windows® Kerberos token (which is a token that contains a claim in which the type is "group membership" and the value is "full-time employees"). The intermediary can then perform block 202 to extract this claim from the token.

[0023] The intermediary can then attempt to map the claim to another claim. In this example, the intermediary (i.e., server 104) can map claims having a type "group membership" and the value "full-time employees" to another claim in which the type is "role" and the value is "purchaser", which is defined for employees that are authorized to make Internet purchases. Thus, for example, claims engine 116 can perform block 204 to determine whether the value of "full-time employees" of type "group membership" can be mapped to the claim having the type "role" and the value "purchaser". This claim can be defined during the configuration of the intermediary. The intermediary can then perform block 206 to form both of these claims into a collection.

[0024] The intermediary can then determine whether to send the purchase request to the on-line retailer using the claim collection. For example, the intermediary can perform block 208 to determine whether the claims associated with the message have a "role" with a value of "purchaser". In this example, John Doe is a "purchaser" and, thus, claims engine 116 will allow the message to be sent to the application (or infrastructure) that sends the purchase order to the on-line retailer.

[0025] Continuing the example, the intermediary can then become the "client" and the on-line retailer can become the "server". The intermediary can send a message (with the purchase order) along with, for example, an X.509 token, which the retailer can then attempt map into a claim of approved customers before processing the purchase order. The message sent by the intermediary can include all or some of the claims resulting from John Doe's message to the intermediary.

[0026] In another embodiment, system 100 operates substantially as described above, but with the additional operation of sending back to client 102 one or more derived claim(s), if any, or location information for where these derived claim(s) are stored in server 104. In some such embodiments, server 104 sends the derived claim(s) to client 102 using a message containing a time-bounded XrML (Extensible Rights Markup Language) license or SAML (Security Assertions Markup Language) assertion, which are both types of security tokens. For example, the message can be conformed by Extensible Markup Language (XML) Version. 1.0 (Third Edition) February 4, 2004 or Security Assertion Markup Language (SAML) Version 1.1 Ratified as Oasis Standard, September 23, 2003. This feature allows the client to include the additional derived claim(s) (or the location information) in the token(s) for subsequent messages. In scenarios having intermediaries, each intermediary can send its derived claims back to the original sender, or alternatively, the endpoint recipient can send all of the derived claims back to the original sender. In this way, performance can be improved by reducing the time/computation needed to perform blocks 204 and 206.

EXEMPLARY CLAIMS ENGINE

[0027] FIG. 3 is a block diagram illustrating exemplary modules of claims engine 116 (FIG. 1), according to one embodiment. In this embodiment, claims engine 116 includes authentication (Auth-N) modules 302_1-302_L, authorization (Auth-Z) modules 306_1-306_M, and an access control module 308. In one embodiment, Auth-N modules 302_1-302_L perform authentication operations (e.g., validates token(s) and/or trusts of token issuer(s)) for "L" different authentication mechanisms. For example, Auth-N modules 302_1-302_L can perform such authentication operations for: (1) X.509

tokens; (2) Kerberos tokens; and (3) Username/Password tokens; as well as other tokens. In addition, in this embodiment, Auth-N modules extract claim(s) from tokens as described above in conjunction with block 202 (FIG. 2).

[0028] Auth-Z modules 306_1-306_M, in one embodiment, perform claim mapping operations such as, for example, mapping extracted claims to other claims for "M" different mappings. For example, Auth-Z modules 306_1-306_M can perform claim mapping operations to: (1) identity claims; (2) group membership claims (e.g., as defined in Windows Server 2003 Authorization Manager); and (3) role claims; as well as other types of claims. In one embodiment, Auth-Z modules 306_1-306_M map claims to other claims as described above in conjunction with block 204 (FIG. 2). In addition, Auth-Z modules 306_1-306_M, can map claims on a per resource basis. For example (continuing the above Internet purchase example), one of the Auth-Z modules can map the aforementioned employee's identity claim (John Doe) to the role claim (purchaser) for a resource (shopping cart: JohnDoe).

[0029] AC module 308, in this embodiment, performs access control operations as described above in conjunction with block 208 (FIG. 2). For example, AC module 308 can make the access control decision (e.g., allow/disallow the message to be further processed) by determining whether the message has an appropriate claim (either extracted or derived) for the resource(s) associated with the message (e.g., whether there is a role claim with value "purchaser" for a "shopping cart" resource).

[0030] In some embodiments, claims engine 116 also extracts resource(s) from the message. In one embodiment, claims engine 116 performs resource extraction before the claim mapping operations performed by the Auth-Z modules. For example, if the message is XML-based and supports XPath (e.g., XML Path Language Version 1.0,

November 16, 1999), AC module 308 can be configured to extract resource(s) contained in the message by evaluating an XPath expression against an XML message. AC module 308 can also be configured to extract a resource at run-time (e.g., determining the identifier for computing platform in which server 104 resides, domain, etc.). Still further, AC module 308 can be configured (e.g., by custom code) to determine a selected property of the message itself (e.g., size in bytes, a count of messages from the sender for the day, or other property that may not be accessible via XPath). Claims engine 116 may also have one or more default resources (e.g., setting a maximum purchase amount).

[0031] FIG. 4 is a data flowchart illustrating an example of data flow in claims engine 116 (FIG. 3), according to one embodiment. Referring to FIGS. 3 and 4, data flows through and is processed by claims engine 116 as follows. Claims engine 116 receives a message 401 from a client that may include one or more token(s). If message 401 does not have at least one token, the data flow essentially aborts. If message 401 does have one or more tokens, the token(s) are then processed by Auth-N modules 302_1-302_L.

[0032] In this example, if message 401 includes an X.509 token, an Auth-N module (e.g., Auth-N module 302_1) performs an authentication process 402_1 that includes authentication operations in accordance with the X.509 Standard and, in addition, claim extraction operations as described above in conjunction with block 202 (FIG. 2). Similarly, if message 401 includes a Kerberos token, an Auth-N module (e.g., Auth-N module 302_2) performs an authentication process 402_2 that includes claim extraction operations and authentication operations in accordance with the Kerberos Standard. Likewise, if message 401 includes a Username/Password token, an Auth-N module (e.g., Auth-N module 302_3) performs an authentication process 402_3 that includes standard

username/password authentication operations and the aforementioned claim extraction operations. In other embodiments, there may be more or less than three Auth-N modules, depending on the scenario.

[0033] The extracted claim(s), if any, are the grouped into a claim collection 404. Claim collection 404 is then operated on by Auth-Z modules 306_1-306_M. As previously described, Auth-Z modules 306_1-306_M map claims into other claims, which may include revoking claim(s).

[0034] In this example, an Auth-Z module (e.g., Auth-Z module 306_1) receives claim collection 404 and performs an authorization process 406_1 in which Auth-Z module 306_1 maps the received claim(s) into Identity claims, as previously described in conjunction with block 204 (FIG. 2). Claim collection 404 then contains the resulting extracted/derived claims.

[0035] Claim collection 404 is then received by another Auth-Z module (e.g., Auth-Z module 306_2) in this example. Auth-Z module 306_2 then performs an authorization process 406_2 to map the extract/derived claims resulting from process 406_1 into Group claims, which will then form part of claim collection 404.

[0036] Similarly, claim collection 404 is next received by another Auth-Z module (e.g., Auth-Z module 306_3) in this example. Auth-Z module 306_3 then performs an authorization process 406_3 to map the extracted/derived claims resulting from processes 406_1 and 406_2 into Role claims. In some embodiments, there may be a custom Auth-Z module (not shown), which performs an authorization process 406_4 to map extracted/derived claims into custom claims (e.g., claims that are specific to application 112).

[0037] Optionally, the claims may undergo authorization processes 406_1-406_3 multiple times to ensure that all of the valid claim mappings have been made, as indicated by a dashed arrow 408 that loops back to authorization process 406_1. As previously described, claims derived by the Auth-Z modules form part of claims collection 404. Claim collection 404 is then used to create message 410, which includes the claim(s) of claim collection 404 with or instead of the token(s). In other embodiments, there may be more or less than three Auth-Z modules, depending on the scenario.

[0038] Alternatively, extracted claims form claim collection 404 while derived claims (derived by the Auth-Z modules) can form another claim collection (not shown). These two claim collections are then used to create message 410, which includes the claim(s) of these claim collections with or instead of the token(s).

[0039] The various embodiments described above may be implemented in computer environments of the server and clients. An example computer environment suitable for use in the server and clients is described below in conjunction with FIG. 5.

[0040] FIG. 5 illustrates a general computer environment 500, which can be used to implement the techniques described herein. The computer environment 500 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer environment 500 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the example computer environment 500.

[0041] Computer environment 500 includes a general-purpose computing device in the form of a computer 502. The components of computer 502 can include, but are not limited to, one or more processors or processing units 504, system memory 506,

and system bus 508 that couples various system components including processor 504 to system memory 506.

[0042] System bus 508 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus, a PCI Express bus, a Universal Serial Bus (USB), a Secure Digital (SD) bus, or an IEEE 1394, *i.e.*, FireWire, bus.

[0043] Computer 502 may include a variety of computer readable media. Such media can be any available media that is accessible by computer 502 and includes both volatile and non-volatile media, removable and non-removable media.

[0044] System memory 506 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 510; and/or non-volatile memory, such as read only memory (ROM) 512 or flash RAM. Basic input/output system (BIOS) 514, containing the basic routines that help to transfer information between elements within computer 502, such as during start-up, is stored in ROM 512 or flash RAM. RAM 510 typically contains data and/or program modules that are immediately accessible to and/or presently operated on by processing unit 504.

[0045] Computer 502 may also include other removable/non-removable, volatile/non-volatile computer storage media. By way of example, FIG. 5 illustrates hard disk drive 516 for reading from and writing to a non-removable, non-volatile magnetic media (not shown), magnetic disk drive 518 for reading from and writing to removable,

non-volatile magnetic disk 520 (*e.g.*, a "floppy disk"), and optical disk drive 522 for reading from and/or writing to a removable, non-volatile optical disk 524 such as a CD-ROM, DVD-ROM, or other optical media. Hard disk drive 516, magnetic disk drive 518, and optical disk drive 522 are each connected to system bus 508 by one or more data media interfaces 525. Alternatively, hard disk drive 516, magnetic disk drive 518, and optical disk drive 522 can be connected to the system bus 508 by one or more interfaces (not shown).

[0046] The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 502. Although the example illustrates a hard disk 516, removable magnetic disk 520, and removable optical disk 524, it is appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the example computing system and environment.

[0047] Any number of program modules can be stored on hard disk 516, magnetic disk 520, optical disk 524, ROM 512, and/or RAM 510, including by way of example, operating system 526, one or more application programs 528, other program modules 530, and program data 532. Each of such operating system 526, one or more application programs 528, other program modules 530, and program data 532 (or some combination thereof) may implement all or part of the resident components that support the distributed file system.

[0048] A user can enter commands and information into computer 502 via input devices such as keyboard 534 and a pointing device 536 (*e.g.*, a "mouse"). Other input devices 538 (not shown specifically) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to processing unit 504 via input/output interfaces 540 that are coupled to system bus 508, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

[0049] Monitor 542 or other type of display device can also be connected to the system bus 508 via an interface, such as video adapter 544. In addition to monitor 542, other output peripheral devices can include components such as speakers (not shown) and printer 546, which can be connected to computer 502 via I/O interfaces 540.

[0050] Computer 502 can operate in a networked environment using logical connections to one or more remote computers, such as remote computing device 548. By way of example, remote computing device 548 can be a PC, portable computer, a server, a router, a network computer, a peer device or other common network node, and the like. Remote computing device 548 is illustrated as a portable computer that can include many or all of the elements and features described herein relative to computer 502. Alternatively, computer 502 can operate in a non-networked environment as well.

[0051] Logical connections between computer 502 and remote computer 548 are depicted as a local area network (LAN) 550 and a general wide area network (WAN) 552. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

[0052] When implemented in a LAN networking environment, computer 502 is connected to local network 550 via network interface or adapter 554. When implemented

in a WAN networking environment, computer 502 typically includes modem 556 or other means for establishing communications over wide network 552. Modem 556, which can be internal or external to computer 502, can be connected to system bus 508 via I/O interfaces 540 or other appropriate mechanisms. It is to be appreciated that the illustrated network connections are examples and that other means of establishing at least one communication link between computers 502 and 548 can be employed.

[0053] In a networked environment, such as that illustrated with computing environment 500, program modules depicted relative to computer 502, or portions thereof, may be stored in a remote memory storage device. By way of example, remote application programs 558 reside on a memory device of remote computer 548. For purposes of illustration, applications or programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of computing device 502, and are executed by at least one data processor of the computer.

[0054] Various modules and techniques may be described herein in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. for performing particular tasks or implement particular abstract data types. These program modules and the like may be executed as native code or may be downloaded and executed, such as in a virtual machine or other just-in-time compilation execution environment. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0055] An implementation of these modules and techniques may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise "computer storage media" and "communications media."

[0056] "Computer storage media" includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer.

[0057] "Communication media" typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. As a non-limiting example only, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

[0058] Reference has been made throughout this specification to "one embodiment," "an embodiment," or "an example embodiment" meaning that a particular

described feature, structure, or characteristic is included in at least one embodiment of the present invention. Thus, usage of such phrases may refer to more than just one embodiment. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0059] One skilled in the relevant art may recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, resources, materials, *etc*. In other instances, well known structures, resources, or operations have not been shown or described in detail merely to avoid obscuring aspects of the invention.

[0060] While example embodiments and applications have been illustrated and described, it is to be understood that the invention is not limited to the precise configuration and resources described above. Various modifications, changes, and variations apparent to those skilled in the art may be made in the arrangement, operation, and details of the methods and systems of the present invention disclosed herein without departing from the scope of the claimed invention.